UNITED STATES PATENT APPLICATION FOR:

METHOD AND APPARATUS FOR PERFORMING BACKUP STORAGE OF CHECKPOINT DATA WITHIN A SERVER CLUSTER

INVENTORS:

PAVAN DEOLASEE
VEERAL SHAH

ATTORNEY DOCKET NUMBER: VRTS 0702

CERTIFICATION OF MAILING UNDER 37 C.F.R. 1.10

I hereby certify that this New Application and the documents referred to as enclosed therein are being deposited with the United States Postal Service on MARCH 29, 2004, in an envelope marked as "Express Mail United States Postal Service", Mailing Label No. EL857899013US, addressed to: Commissioner for Patents, Mail Stop PATENT APPLICATION, P.O. Box 1450, Alexandria, VA 22313-1450

Kathleen Faughnan
Signature

Kathleen Faughnan
Name

March 29, 2004
Date of signature

MOSER, PATTERSON & SHERIDAN, LLP
595 Shrewsbury Ave.
Shrewsbury, New Jersey 07702
(732) 530-9404

256619

PATENT
Attorney Docket No.: VRTS 0702

METHOD AND APPARATUS FOR PERFORMING BACKUP STORAGE OF CHECKPOINT DATA WITHIN A SERVER CLUSTER

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] Embodiments of the present invention generally relate to data storage systems, and more particularly, to a method and apparatus for performing backup storage of checkpoint data.

Description of the Related Art

[0002] Modern computer networks generally comprise a plurality of user computers connected to one another and to a computer server via a communications network. To provide redundancy and high availability of information and applications that are executed upon a computer server, multiple computer servers may be arranged in a cluster, i.e., forming a server cluster. Such server clusters are available under the trademark VERITAS CLUSTER SERVER from Veritas Software Corporation of Mountain View, California. In a server clusters, the plurality of servers communicate with one another to facilitate failover redundancy such that when software or hardware (i.e., computer resources) become inoperative on one server, another server can quickly execute the same software that was running on the inoperative server substantially without interruption. As such, a user of services that are supported by a server cluster would not be substantially impacted by an inoperative server or software.

[0003] To facilitate the substantially seamless transition of user service to another server within the server cluster, the production server, i.e., the server that is presently supporting users of the server services, stores checkpoint data in random access memory (RAM). This checkpoint data is essentially the data being used by software at particular times as well as the server state at those particular times. The backup software within the production server takes a "snapshot" of the data and the server state, then stores that information as checkpoint data. To create redundancy, the checkpoint data is remotely stored on a backup server, i.e., another server in the

server cluster. Upon failure of the software or production server, the software is booted on the backup server and the checkpoint data can be used to start the software at approximately the position within the software where the failure occurred.

[0004] Upon failover, the backup server becomes the production server from the view of the user without substantial interruption of the software utilization. Thus, upon failover to the backup server, the software is executed from the last saved state which can then use the stored data related to that saved state.

[0005] The storage of the checkpoint data on a single backup server or a node within the server cluster can be problematic if the production server and the backup server fail simultaneously.

[0006] Therefore, there is a need in the art for a method and apparatus for improving the fault tolerance of the storage of checkpoint data within a server cluster.

SUMMARY OF THE INVENTION

[0007] The embodiments of the present invention are generally directed to a method and apparatus for storing checkpoint data in a fault tolerant manner. To facilitate redundancy, the invention distributes the checkpoint data across a plurality of backup servers. In one embodiment, the method distributes the checkpoint data by striping the data across a plurality of backup servers for storage in memory such as random access memory (RAM), disk drive storage, and the like. Specifically, the invention subsegments each segment of checkpoint data produced by a production server, produces parity data for parity groups of the subsegmented checkpoint data, and stripes the subsegments and the parity data across a number of backup servers within a server cluster for storage therein. In other embodiments of the invention, the checkpoint data may be distributed in other ways, e.g., mirroring, striping without parity, and the like. The checkpoint data distribution may be performed by the production server, a backup server, or a computer specifically tasked to perform checkpoint data distribution. Upon failure of the production server, one of the backup servers reassembles the segments of checkpoint data using the

subsegments and/or the parity data. One of the backup servers will then execute software using the reassembled checkpoint data.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The following detailed description makes reference to the accompanying drawings which are now briefly described.

[0009] Figure 1 is a block diagram of a computer network having a server cluster that operates in accordance with the present invention;

[0010] Figure 2 depicts a flow diagram of a method of operating a server cluster in accordance with the present invention;

[0011] Figure 3 graphically depicts a diagram of data segmentation and storage within a server cluster in accordance with one embodiment of the present invention; and

[0012] Figure 4 depicts a flow diagram of a method for utilizing checkpoint data that is stored in accordance with the present invention.

[0013] While the invention is described herein by way of example using several embodiments and illustrative drawings, those skilled in the art will recognize that the invention is not limited to the embodiments of drawing or drawings described. It should be understood that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the invention is to cover all modification, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims. The headings used herein are for organizational purposes only and are not meant to be used to limit the scope of the description or the claims. As used throughout this application, the word "may" is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words "include," "including," and "includes" mean including, but not limited to.

DETAILED DESCRIPTION OF THE INVENTION

[0014] Figure 1 depicts a computer network 100 in which one embodiment of the present invention may be utilized. The invention, as shall be discussed below, is a method and apparatus for storing checkpoint data that is created in a production server 108 upon a plurality of backup servers 110_1, 110_2, 110_3 ... 110_n. As such, the checkpoint data is stored in a redundant, fault-tolerant manner.

[0015] The computer network 100 comprises a plurality of client computers 102_1, 102_2 ... 102_n that are connected to a network 104. The client computers 102 may contain one or more individual computers, wireless devices, personal digital assistants, desktop computers, laptop computers or any other digital device that may benefit from connection to a computer network.

[0016] The computer network 104 is a conventional computer network which may be an Ethernet network, a fiber channel network and the like. The client computers 102 may be connected to a server cluster 106 through a firewall, a router, or some form of switch (not shown). The server cluster 106 generally comprises multiple servers 108, 110_1, 110_2, ... 110_n that are interconnected to one another by a server cluster network 138. Although the servers within the server cluster 106 may be identical, one of the servers will be operating to service particular clients (users) at any particular time. This server is identified as the production server 108 while all other servers 110_1, 110_2 ... 110_n are deemed backup servers vis-à-vis the production server 108. All the servers 108, 110_1, 110_2 ... 110_n may be referred to as nodes of the server cluster 106. Although one server cluster 106 is depicted in Figure 1, those skilled in the art will realize that many server cluster benefiting from the invention can be connected to the computer network 104. Such a server cluster 106 may be a VERITAS CLUSTER SERVER available from Veritas Software Corporation of Mountain View, California.

[0017] The production server 108 generally includes at least one central processing unit (CPU) 112, support circuits 114 and memory 116. Additionally, the production server 108 may contain mass storage 124 or the server 108 may be connected to a common mass storage system (not shown) that is shared by a plurality of servers.

The CPU 112 may include one or more commercially available processors. The support circuits 114 are well known circuits that include cache, power supplies, clocks, input/output interface circuitry, and the like.

[0018] The memory 116 may include random access memory, read only memory, removable disk memory, flash memory, and various combinations of these types of memory. The memory 116 is sometimes referred to as main memory and may in part be used as cache memory. The memory 116 stores checkpoint data 119 and may store various software applications such as application software 118 and backup software 120. In one embodiment of the invention, the backup software 120 further comprises a checkpoint data backup module 122 and a distribution module 136. In other embodiments of the invention, modules 122 and 136 may be stored and executed by a backup server 100 or a specialized computer (not shown) that is solely used to coordinate backup of the checkpoint data 119.

[0019] The backup software 120 is used to insure that the checkpoint data 119 as well as other data that is stored in the production server 108 is sent to the backup servers 110_1, 110_2 ... 110_n for remote storage from the production server 108. Such backup software 120 insures that the production server 108 is fault tolerant such that a failure of the production server 108 will quickly be compensated by one or more of the backup servers 110_1, 110_2 ... 110_n. The checkpoint data backup module 122 stores checkpoint data 119 at various points during execution of the application software. As one or more applications 118 are being executed by the production server 108, the checkpoint data backup module 122 will intermittently save the state information of the application as well as data that is used by the application at that point in time. In this manner, the state and data are related at the particular point in time that the checkpoint data is stored. The invention, as will be described in more detail below, uses a distribution module 136 to subsegment the checkpoint data 119 as well as form parity information from a plurality of checkpoint data subsegments. The distribution module 136 then sends the subsegments and parity information to various backup servers 110_1, 110_2 ... 110_n for storage in a diversified manner. In one embodiment, the diversified manner involves striping subsets of the checkpoint data and the parity information on memory devices within each of the backup

servers 110_1, 110_2 ...110_n. The checkpoint data may be stored in the backup servers in RAM, disk drive storage, and the like.

[0020] Each backup server 110_1, 110_2 ... 110_n comprises a CPU 126, support circuits 128, memory 130 and mass storage device 132. As discussed previously, the mass storage device 132 may be a mass storage system that supports a plurality of the backup servers 110. The checkpoint data 134 is stored in, for example, the memory 130 for recall and use when the production server 108, corresponding to that checkpoint data, fails. Upon failover, the backup servers 110_1, 110_2 ... 110_n cooperate to execute the applications that were running on the production server from the server state indicated by the checkpoint data. By storing the checkpoint data 134 in the memory 130, the checkpoint data 134 may be recalled and used quickly by the backup server 110. Alternatively, the checkpoint data 134 can be stored upon the mass storage device 132, e.g., a disk drive or disk drive array, as indicated by the dashed box.

[0021] To facilitate communications within the server cluster, the backup servers and the production server are interconnected by, for example, a private network 138. The private network 138 facilitates redundant use of the servers by providing communications between each of the servers within the server cluster 106. As such, the failover process is coordinated amongst the servers in the cluster via the network 138.

[0022] Figure 2 depicts a flow diagram of a method 200 of operation for the checkpoint data backup module 122. The function of the distribution module 136 is embedded within the method 200.

[0023] The method 200 starts at step 202 and proceeds to step 204, wherein the method 200 identifies and accesses the checkpoint data in the production server. Generally, the checkpoint data is stored in random access memory within the production server. The method 200 will typically request the checkpoint data after each update of the data by its associated application. At step 206, the checkpoint data is sent to the distribution module. The distribution module may distribute the

checkpoint data in one of a number of ways. For example, the data could be: (1) replicated for storage on a plurality of backup servers (e.g., a mirroring technique), (2) striped across the backup servers, (3) striped with parity data across the backup servers, and the like.

[0024] In one specific embodiment, at step 208, within the distribution module, the method 200 divides the checkpoint data into subsegments. As such, a segment of checkpoint data containing the production server state, as well as any data that is involved in the execution of the application at that state, is divided into subsegments. At step 210, the distribution module selects groups of subsegments (e.g., three subsegments to a redundancy group) and calculates the parity data related to the redundancy groups. For example, if a segment of checkpoint data is divided into 9 segments labeled 0-8, then parity data will be generated for every three subsegments (see FIG. 3 below). At step 212, the subsegments and the parity data are sent to a plurality of backup servers in a striped manner. By striped manner, it is meant that the data is distributed over and stored in a plurality of memory devices coupled to or embedded within the backup servers. Such distribution of the data in this manner provides redundancy and fault tolerance for the stored information. One specific embodiment that is used for striping data is described with reference to Figure 3 below. At step 214, the subsegments are stored along with the parity data on the plurality of backup servers. The method 200 ends at step 216. The method 200 is executed after each checkpoint such that when the checkpoint data and the state of the executing application are known (i.e., updated), the method 200 executes and stores the checkpoint data and parity information on the various backup servers within the server cluster.

[0025] Figure 3 graphically depicts one embodiment of an illustrative striping algorithm that is used to distribute checkpoint data across the backup servers. Each segment 300 of checkpoint data is divided into N subsegments, where N is an integer that is greater than 0. In the depicted embodiment, N is 9 and the subsegments 0-8. To stripe the data across four backup servers, the subsegments are separately stored in memory as groups 302, 304, 306 and 308. Subsegment 0 and subsegment 8 are stored in memory 130_1 of the first backup server.

Additionally, parity data for subsegments 3, 4 and 5, i.e., the exclusive OR of the data in subsegments 3, 4 and 5, is also stored in memory 130_1 as group 302. Group 304, comprising subsegment 1 and subsegment 3 and the parity data of subsegments 6, 7 and 8, is stored in memory 130_2. Memory 130_3 coupled to a third backup server stores the subsegment group 306 comprising subsegments 2, 4 and 6. Lastly, memory 130_4 coupled to the fourth backup server stores group 308 comprising parity data for subsegments 0, 1 and 2 and subsegment 5 and subsegment 7.

[0026] In this manner, the parity data and the subsegments are striped across four memory devices within four separate backup servers. Each segment of checkpoint data is processed in this manner until all of the data is distributed across a number of backup servers.

[0027] Upon failure of the production server and up to one of the backup servers, the application may be restarted using the checkpoint data at the point at which the failure occurred. If both the production server and one of the backup servers fail, the parity data is used to reconstruct each checkpoint data segment. The striping of the data which is similar to that used in RAID 5 in a redundant fault tolerant disk drive array may also be striped and/or stored in other redundant manners either with or without parity data. For example, the checkpoint point data could be mirrored onto a plurality of storage devices coupled to separate backup servers (similar to RAID 1 or 2), or the data may be striped without parity data (similar to RAID 3), and the like.

[0028] Figure 4 depicts a method 400 that summarizes operation of the server cluster upon failure of the production server 108. The method 400 starts at step 402 (e.g., upon failure of the production server) and proceeds to step 404 wherein the method confirms that the production server has failed. The conventional failover process is initiated to cause failover to a backup server. As such, one of the backup servers will begin to execute the software that was running on the production server. To facilitate use of the software, the backup server will require the checkpoint data from the last checkpoint before the production server failure. At step 408, the method 400 accesses the checkpoint data subsegments and parity

data that are stored in the backup servers. As such, the backup server that is now operating as the new production server, requests the checkpoint data (including parity) from the backup servers. At step 410, the method 400 computes and stores the checkpoint data into the new production server, (i.e., the data segments are reformed). At step 412, the new production server continues software execution (i.e, supporting users) using the checkpoint data from the point of failure onwards. The method ends at step 414.

[0029] Although the foregoing discussion assumes that the production server contains backup software that use a diversification module for subsegmenting the checkpoint data, that module may be located in one or more of the backup servers such that the production server will send the diversification checkpoint data to the backup server that is, at that moment, executing a diversification module for storing the checkpoint data in a striped manner. In other embodiments, there may be a separate server or computer for handling the diversification of the checkpoint data across the backup servers.

[0030] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.